MULTIMEDIA UNIVERSITY®

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2015/2016

TCE 3411/TSN 3151 – PARALLEL PROCESSING

(All sections / Groups)

4th MARCH 2016
9.00 a.m - 11.00 a.m
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This question paper consists of 4 pages only (including this page).
2. **This is an open-book exam,** so the questions are more in-depth and there are fewer questions than normal exams.
3. There are **THREE (3) QUESTIONS** in this paper. **Answer ALL questions.** All questions carry equal marks (20 marks) and the distribution of the marks for each question is given.
4. All questions should be answered using the C programming language.
5. Write your answers in the Answer Booklet provided.
6. State all assumptions clearly.

Question 1 [20 marks]

a) One way to approximate π is to pick random points in an N by N square of the plane, and compute what percentage of the random points all within the circle of diameter N that has the same center as the square.

Write an **OpenMP** program that computes a `double` approximation of π using 10,000,000 random points, where a random point is a pair of random `ints`. Assume a function `random` that produces a random `int` between `MAX_INT` and `MAX_INT`.

Your program should work with any number of threads P and obtain good speedup when `P is much smaller than 10,000,000` (assuming that `random` is independent for each thread, so calling random does not imply synchronization).

[10 marks]

b) Implement an MPI program that takes `N ints` and produces `N double` approximations of π, each using as many iterations as specified by the corresponding input integer `int`. Assume a function `read.input.data` that takes an array of `ints` and fills it with `N` numbers read from a file that is accessible only to process 0.

You can re-use any helper functions that you defined for Question 1(a). Assume that integers in the input are evenly distributed (as opposed to having many more small numbers than large numbers in a given section of the input array), and assume that the number of processes `P` evenly divides `N`.

[10 marks]

Continued.....

Question 2 [20 marks]

a) Given an array of *n* randomly generated integers (within the range 0-10), the goal of the program is to find the difference between the total of the numbers divisible by 4 and the total of the numbers divisible by 3. Based on sample output provided below, write the MPI Program and utilize the `MPI_Reduce` routines to compute the final results.

```
e.g.
Assume that the input:
n = 20
a[] = {1, 6, 2, 6, 3, 1, 4, 8, 5, 2, 0, 3, 8, 9, 5, 2, 4, 7, 3, 4}

The output:
Total of the numbers divisible by 4 = 4 + 8 + 0 + 8 + 4 + 4 = 28
Total of the numbers divisible by 3 = 6 + 6 + 3 + 0 + 3 + 9 + 3 = 30

Difference = 30-28 = 2
```

[10 marks]

b) Write the program in C/C++ using Pthreads.

[10 marks]

Continued.....

Question 3 [20 marks]

Given the serial program below, Pi (π) is calculated using an integral approximation method (as explained inside the program comments).

(a) Modify the program into a parallel **Message Passing Interface (MPI)** version in Single Program Multiple Data (SPMD) form

[20 marks]

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define f(x) ((double)(4.0/(1.0+x*x)))
#define pi ((double)(4.0*atan(1.0)))
int main ()
{
/* This simple program approximates pi by computing pi = integral from 0
to 1 of 4/(1+x*x)dx which is approximated by sum from k=1 to N of
4/(1+((k-.5)/N)**2). The only input data required is N. */

      double err, sum, w;
      int i, N;
      FILE *GetInterval;

      GetInterval = fopen( "./values", "r" );
      fscanf(GetInterval, "%d", &N);
      printf("Approximation interval is %d\n", N);

      while (N > 0)
      {
            w = 1.0/(double)N;
            sum = 0.0;
            for (i = 1; i <= N; i++)
            {
              sum = sum + f(((double)i-0.5)*w);
            }
            sum = sum * w;
            err = sum - pi;
            printf("sum, err = %7.5f, %10e\n", sum, err);
            fscanf(GetInterval, "%d", &N);
            printf("Approximation interval is %d\n", N);
      }
      fclose (GetInterval);
      return 0;
}
```

END OF EXAM PAPER